

OCR A Level

Computer
Science

H446 – Paper 1



Arrays, tuples and records

Unit 7
Data structures



PG ONLINE

Objectives

- Be familiar with the concept of a data structure
- Understand how data is represented and stored within different structures including
 - arrays up to three dimensions
 - tuples
 - records
- Use 1- 2- and 3-dimensional arrays in the design of solutions to simple problems

Variable declarations

- Describe the function of a variable declaration
- Why do we use them?
- Do all languages have explicit variable declarations?

Data structures

- In programming languages there are elementary data types such as char, real, integer and Boolean
- There may be built-in structured data types such as strings, arrays, lists and records
- Structured data types are usually made up of elementary data types
- What is a string made from?
- What would an array of ages look like?

An array of names

- If you need to sort, for example, 100 names of towns into alphabetical order, it is inconvenient to have different names town1, town2, town3 to hold them
- Instead, they are all given one name, and referred to by an **index** as, for example, **town[0], town[1] ...town[n]**
 - Note the use of [] to hold the index
 - Some programming languages may use () instead



Referencing array elements

Follow the pseudocode and complete the trace table

```
name1 = "Joe"  
name2 = "Jim"  
name[0] = "Moe"  
name[1] = "Mae"  
name[2] = "Mic"  
for i = 0 to 2  
    print (name[i])  
next i  
print (name1)  
print (name2)
```

name						
name1	name2	0	1	2	i	OUTPUT



Referencing array elements

Follow the pseudocode and complete the trace table

```
name1 = "Joe"  
name2 = "Jim"  
name[0] = "Moe"  
name[1] = "Mae"  
name[2] = "Mic"  
for i = 0 to 2  
    print (name[i])  
next i  
print (name1)  
print (name2)
```

name						
name1	name2	0	1	2	i	OUTPUT
Joe	Jim	Moe	Mae	Mic	0	Moe
					1	Mae
					2	Mic
						Joe
						Jim



Arrays of numbers

- Arrays can also hold numbers
 - Assume the inputs are: 60, 70, 80
- What does this algorithm do?

```
total = 0
for i = 0 to 2
    mark[i] = input
    total = total + mark[i]
next i
avg = total / 3
print (avg)
print ("3rd mark", mark[2])
```

mark						
0	1	2	i	total	avg	OUTPUT

Arrays of numbers

- Arrays can also hold numbers
 - Assume the inputs are: 60, 70, 80
- What does this algorithm do?

```
total = 0
for i = 0 to 2
    mark[i] = input
    total = total + mark[i]
next i
avg = total / 3
print (avg)
print ("3rd mark", mark[2])
```

mark						
0	1	2	i	total	avg	OUTPUT
				0		
			0			
60				60		



Arrays of numbers

- Arrays can also hold numbers
 - Assume the inputs are: 60, 70, 80
- What does this algorithm do?

```
total = 0
for i = 0 to 2
    mark[i] = input
    total = total + mark[i]
next i
avg = total / 3
print (avg)
print ("3rd mark", mark[2])
```

mark						
0	1	2	i	total	avg	OUTPUT
				0		
			0			
60				60		
	70		1	130		

Arrays of numbers

- Arrays can also hold numbers
 - Assume the inputs are: 60, 70, 80
- What does this algorithm do?

```
total = 0
for i = 0 to 2
    mark[i] = input
    total = total + mark[i]
next i
avg = total / 3
print (avg)
print ("3rd mark", mark[2])
```

mark						
0	1	2	i	total	avg	OUTPUT
				0		
			0			
60				60		
	70		1	130		
		80	2	210		



Arrays of numbers

- Arrays can also hold numbers
 - Assume the inputs are: 60, 70, 80
- What does this algorithm do?

```
total = 0
for i = 0 to 2
    mark[i] = input
    total = total + mark[i]
next i
avg = total / 3
print (avg)
print ("3rd mark", mark[2])
```

mark						
0	1	2	i	total	avg	OUTPUT
				0		
			0			
60				60		
	70		1	130		
		80	2	210		
					70	



Arrays of numbers

- Arrays can also hold numbers
 - Assume the inputs are: 60, 70, 80
- What does this algorithm do?

```
total = 0
for i = 0 to 2
    mark[i] = input
    total = total + mark[i]
next i
avg = total / 3
print (avg)
print ("3rd mark", mark[2])
```

mark						
0	1	2	i	total	avg	OUTPUT
				0		
			0			
60				60		
	70		1	130		
		80	2	210		
					70	
						70
						3rd mark 80

Worksheet 1

- Complete the questions in **Task 1**



2D arrays

- It is also possible to have two-dimensional arrays
- What values would appear in the table once the algorithm has run?

		j	
		0	1
i	0	?	?
	1	?	?

```
array grid[2,2]
for i = 0 to 1
    for j = 0 to 1
        grid[i,j] = "x"
    next j
next i
grid[0,1] = "y"
```

2D arrays

- It is also possible to have two-dimensional arrays
- What values would appear in the table once the algorithm has run?

		j	
		0	1
i	0	x	y
	1	x	x

```
array grid[2,2]
for i = 0 to 1
    for j = 0 to 1
        grid[i,j] = "x"
    next j
next i
grid[0,1] = "y"
```


2D array trace table

- Fill in the trace table if the user inputs
a, b, c, x, y, z
- Don't forget to fill in i and j

```
array letters[2,3]
for i = 0 to 1
  for j = 0 to 2
    letters[i, j] = input
  next j
next i
```

letters							
[0,0]	[0,1]	[0,2]	[1,0]	[1,1]	[1,2]	i	j

2D array trace table

- Fill in the trace table if the user inputs
a, b, c, x, y, z
- Don't forget to fill in i and j

```
array letters[2,3]
for i = 0 to 1
  for j = 0 to 2
    letters[i, j] = input
  next j
next i
```

letters							
[0,0]	[0,1]	[0,2]	[1,0]	[1,1]	[1,2]	i	j
						0	
a							0

2D array trace table

- Fill in the trace table if the user inputs
a, b, c, x, y, z
- Don't forget to fill in i and j

```
array letters[2,3]
for i = 0 to 1
  for j = 0 to 2
    letters[i, j] = input
  next j
next i
```

letters							
[0,0]	[0,1]	[0,2]	[1,0]	[1,1]	[1,2]	i	j
						0	
a							0
	b						1
		c					2

2D array trace table

- Fill in the trace table if the user inputs
a, b, c, x, y, z
- Don't forget to fill in *i* and *j*

```

array letters[2,3]
for i = 0 to 1
  for j = 0 to 2
    letters[i, j] = input
  next j
next i
  
```

letters							
[0,0]	[0,1]	[0,2]	[1,0]	[1,1]	[1,2]	i	j
						0	
a							0
	b						1
		c					2
						1	
			x				0



2D array trace table

- Fill in the trace table if the user inputs
a, b, c, x, y, z
- Don't forget to fill in *i* and *j*

```
array letters[2,3]
for i = 0 to 1
  for j = 0 to 2
    letters[i, j] = input
  next j
next i
```

letters							
[0,0]	[0,1]	[0,2]	[1,0]	[1,1]	[1,2]	i	j
						0	
a							0
	b						1
		c					2
						1	
			x				0
				y			1
					z		2

Worksheet 1

- Complete **Task 2**



Multi-dimensional arrays

- It is possible to define arrays with any number of dimensions
- The array is a set of elements with the same data type
 - For example, you could have a 3-dimensional array which would hold x, y and z coordinates in 3-D space
 - You could use a 4-dimensional array to hold sales for branches of a supermarket, by country, store, department, and year
 - e.g. `sales[c, s, d, y] = 23450`

Tuples

- A tuple has the following properties:
 - It is an ordered set of values
 - It may have elements of mixed types (string, integer, real, Boolean)
 - It is **immutable**, meaning the elements of a tuple cannot change
- Example:
 - Suppose a tuple has elements
PlayerID, Lastname, Firstname, score1, score2, score3
 - Player1 = (34296, "Jones", "Jane", 125, 150, 137)

Tuples

- Tuples can be put together in a 1D array

[1]	(34296, "Jones", "Jane", 125, 150, 137)
[2]	(115539, "Fox", "Fred", 150, 154, 146)

- Why is this data structure not a 2D array?

[1]	(34296, "Jones", "John", 125, 150, 137)
[2]	(115539, "Fox", "Fred", 150, 154, 146)
[3]	(52387, "Smith", "Sam")
...	
[n]	(227538, "Zimmer", "Zoe", 166, 178, 123, 156)

Immutability of tuples

- In the example given:

PlayerID, Lastname, Firstname, score1, score2, score3

- Player1 = (34296, "Jones", "Jane", 125, 150, 137)
- If Jane's name has been misspelt, and should be spelt "Jayne", the tuple cannot be changed
- It is invalid to write, for example,
 - Player1[2] = "Jayne"
- Nor can you cannot add an extra element to a tuple

Records

- Records are composed of a fixed number of fields of different data types
 - Conceptually, they resemble a spreadsheet

ID	Lastname	Firstname	Dept
2468	Jones	John	243
3579	Smith	Sam	634
1428	Zimmer	Zoe	243

Records

- A record can be implemented as an object
- A record can be treated like a tuple
- In writing files, each record is usually written as a single line

ID	Lastname	Firstname	Dept
2468	Jones	John	243
3579	Smith	Sam	634
1428	Zimmer	Zoe	243

Static and Dynamic

- Data structures are characterised according to their ability to grow and shrink on demand
- Dynamic data structures change size when required
 - Lists (arraylists) grow when items are added and shrink when items are removed
- Static data structures cannot change size
 - Arrays in most languages do not change size automatically
 - Some languages provide functions to resize, but this has to be programmed, it is not done automatically

Worksheet 1

- Complete **Task 3**



Plenary

- Complete the table to show the characteristics of these different data structures

	Array	Tuple	Record
Fixed size (static)			
Number of items may grow and shrink			
All items must be of same data type			
Fixed number of items			
Can have 1 or more dimensions			
Items may be of different data types			
Can hold an item of the other two structures			



Plenary

- Complete the table to show the characteristics of these different data structures

	Array	Tuple	Record
Fixed size (static)	<input type="checkbox"/>	<input type="checkbox"/>	
Number of items may grow and shrink			
All items must be of same data type	<input type="checkbox"/>		
Fixed number of items	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Can have 1 or more dimensions	<input type="checkbox"/>		
Items may be of different data types		<input type="checkbox"/>	<input type="checkbox"/>
Can hold an item of the other two structures	<input type="checkbox"/>		

Copyright

© 2016 PG Online Limited

The contents of this unit are protected by copyright.

This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it are supplied to you by PG Online Limited under licence and may be used and copied by you only in accordance with the terms of the licence. Except as expressly permitted by the licence, no part of the materials distributed with this unit may be used, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic or otherwise, without the prior written permission of PG Online Limited.

Licence agreement

This is a legal agreement between you, the end user, and PG Online Limited. This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it is licensed, not sold, to you by PG Online Limited for use under the terms of the licence.

The materials distributed with this unit may be freely copied and used by members of a single institution on a single site only. You are not permitted to share in any way any of the materials or part of the materials with any third party, including users on another site or individuals who are members of a separate institution. You acknowledge that the materials must remain with you, the licencing institution, and no part of the materials may be transferred to another institution. You also agree not to procure, authorise, encourage, facilitate or enable any third party to reproduce these materials in whole or in part without the prior permission of PG Online Limited.

